

CLAIMS

- 1 1. A method for efficiently parsing input data, comprising:
2 receiving a data file;
3 retrieving a stored version of the data file and a template/token tree
4 corresponding to the data file, the tree including at least one static node;
5 comparing the stored version of the data file with the received data file to
6 identify non-matching content in the received data file;
7 parsing only the non-matching content to form subtrees;
8 creating a mapping from the template/token tree to the subtrees.
- 1 2. The method of claim 1 wherein the step of creating the mapping from the tree
2 to the subtrees further comprises:
3 replacing at least one static node of the template/token tree with a token; and
4 creating a mapping from each token to at least one subtree.
- 1 3. The method of claim 1 wherein creating the mapping from the tree to the
2 subtrees further comprises:
3 adding at least one token node to the template/token tree; and
4 creating a mapping from each token to at least one subtree.
- 1 4. The method of claim 1 wherein the data file is a web page.
- 1 5. The method of claim 1 wherein the data file is an HTML file.
- 1 6. A method for efficiently parsing web pages, comprising:
2 receiving a first HTML page;

3 retrieving a cached version of the HTML page and a template/token tree
 4 corresponding to the first HTML page, the tree including at least one static
 5 node;
 6 comparing the cached version of the HTML page with the received HTML page
 7 to identify non-matching content in the received HTML page;
 8 parsing only the non-matching content to form at least one subtree;
 9 creating a mapping from the template/token tree to the subtrees.

1 7. The method of claim 6 wherein creating the mapping from the tree to the
 2 subtrees further comprises:
 3 replacing at least one static node of the template/token tree with a token; and
 4 creating a mapping from each token to at least one subtree.

1 8. A method for efficiently parsing HTML pages, comprising:
 2 receiving a first HTML page;
 3 responsive to a determination that a cached version of the HTML page exists:
 4 retrieving the cached version of the HTML page and a first
 5 template/token tree corresponding to the first HTML page, the
 6 first tree including at least one static node;
 7 comparing the cached version of the first HTML page with the
 8 received HTML page to identify non-matching content;
 9 parsing only the non-matching content to form a subtree;
 10 associating the first tree and the subtree;
 11 responsive to a determination that the cached version of the HTML page does
 12 not exist:
 13 parsing the received HTML page to form a second template/token
 14 tree, the second tree containing at least one static node; and

15 storing the second tree and the received HTML page.

- 1 9. A method for providing derivative services comprising:
2 receiving a first HTML page;
3 constructing a template/token tree from the received HTML page, the tree
4 comprising a plurality of nodes;
5 determining that at least one node of the tree contains static content;
6 determining that at least one node of the tree contains dynamic content;
7 replacing the nodes of the tree containing dynamic content with tokens;
8 parsing the dynamic content to form subtrees; and
9 mapping the tokens to the subtrees.
- 1 10. A method of providing derivative services, comprising:
2 receiving a request for derivative services content from a customer;
3 retrieving data from a plurality of primary service providers on behalf of the
4 customer, by:
5 identifying static content that has been previously retrieved from the
6 primary service providers and stored, and corresponding
7 template/token trees that have also been stored;
8 identifying dynamic content that differs from the previously retrieved
9 content;
10 parsing the dynamic content to form subtrees;
11 adding tokens to the template/token trees;
12 mapping the tokens to the subtrees;
13 creating at least one content page comprising the retrieved data; and
14 providing the created pages to the customer.

1 11. A method for efficiently parsing input data, comprising:
 2 receiving a first data file;
 3 retrieving a stored template/token tree, the stored template/token tree having
 4 content associated with the first data file and containing at least one static
 5 node and at least one token;
 6 retrieving a second data file, the second data file associated with the first data
 7 file;
 8 identifying non-matching content present only in the first data file;
 9 parsing only the non-matching content of the first data file to form at least one
 10 subtree; and
 11 mapping at least one of the tokens to at least one of the subtrees.

1 12. The method of claim 11, further comprising:
 2 responsive to identifying non-matching content present only in the first file:
 3 adding at least one new token to the template/token tree.

1 13. A system for efficiently parsing input data, comprising:
 2 at least one virtual browser for retrieving content from primary content servers;
 3 an identification engine, communicatively coupled to the virtual browser for
 4 identifying retrieved content;
 5 a cache, communicatively coupled to the virtual browser and the parsing engine,
 6 for storing retrieved content and template/token trees;
 7 a comparison engine, coupled to the virtual browser for comparing retrieved
 8 content with stored content to identify differing content not stored in the
 9 cache;

10 a parsing engine, communicatively coupled to the virtual browser, for parsing
11 content identified by the comparison engine as differing content and forming
12 subtrees from the content; and
13 a content server, coupled to the virtual browser.

1 14. The system of claim 13, further comprising a token master, coupled to the
2 cache, for allocating new tokens to the virtual browser.

TO: "332920"